CLAIMS

What is claimed is:

1. In a computer system having a plurality of threads, including a first and second thread, a method of executing more than one thread at a time, the method comprising:

providing a first and a second reorder buffer;

reading first instructions and first operands associated with the first thread from the first reorder buffer;

executing one of the first instructions and storing a result in the first reorder buffer, wherein storing the result includes marking the result with a tag associating the result with the first thread;

reading second instructions and second operands associated with the second thread from the second reorder buffer; and

executing one of the second instructions and storing a result in the second reorder buffer, wherein storing the result includes marking the result with a tag associating the result with the second thread.

2. The method of claim 1, wherein the method further includes providing a first execution unit, and wherein executing one of the first or one of the second instructions includes executing in the first execution unit.

3. The method of claim 2, wherein the first execution unit includes a floating point unit.

4. The method of claim 3, the method further including providing a second execution unit, and wherein executing one of the first or one of the second instructions includes executing in the second execution unit.

5. The method of claim 4, wherein the second execution unit includes a memory order buffer.

6. The method of claim 5, the method further including providing an instruction pipeline.

7. The method of claim 6, wherein executing one of the second instructions includes allocating the first and second execution units to the second instruction if the first thread stalls in the instruction pipeline.

8. The method of claim 6, wherein executing one of the second instructions includes allocating the first and second execution units to the second instruction if the first thread is flushed from the instruction pipeline.

9. The method of claim 6, wherein the method further includes:

providing a first and second instruction fetch/decode units, wherein the first instruction fetch/decode unit is associated with the first thread and the second instruction fetch/decode unit is associated with the second thread; and

spacing the instruction fetch/decode units evenly around the instruction pipeline.

10. The method of claim 6, wherein executing one of the first or one of the second instructions includes the first and second threads prioritizing instructions within its thread.

11. The method of claim 10, wherein prioritizing instructions includes increasing a priority of an instruction when the instruction completes a loop around the instruction pipeline.

12. The method of claim 10, wherein an instruction is assigned a priority during compilation based on a dependency of other instructions.

13. A processor comprising:

an instruction pipeline; and

a results pipeline, wherein the instruction pipeline and the results pipeline are counter rotating queues;

a first execution unit in communication with the results and instruction pipelines;

a plurality of threads including a first and second thread;

a first and a second reorder buffer, the first reorder buffer associated with the first thread and the second reorder buffer associated with the second thread; and

a first instruction fetch/decode unit.

14. The processor of claim 13, further including a second instruction fetch/decode unit, wherein the first instruction fetch/decode unit is associated with the first thread and the second instruction fetch/decode unit is associated with the second thread.

15. The processor of claim 14, wherein the first and second fetch/decode units are spaced evenly at first and second locations around the instruction and results pipelines.

16. The processor of claim 13, wherein instructions are multiplexed from the first instruction fetch/decode unit into the first and second threads.

17. The processor of claim 16, wherein instructions from the first fetch/decode unit are multiplexed into points spaced evenly at first and second locations around the instruction and results pipelines.

18. The processor of claim 17, further including a second execution unit, wherein the first and second execution unit are spaced evenly around the instruction and results pipeline.

19. The processor of claim 13, wherein instructions include a tag associating the instruction with a thread.

20. The processor of claim 13, wherein the pipeline is two instructions wide.